AD-758 162

A RESOURCE SHARING EXECUTIVE FOR THE ARPANET

Robert H. Thomas

Bolt Beranek and Newman, Incorporated

Prepared for:

Advanced Research Projects Agency

March 1973

BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

BBN Report No. 2522 March 1973

A Resource Sharing Executive for the ARPANET*

by Robert H. Thomas
Bolt, Beranek and Newman, Inc.
Cambridge, Mass., 02138

This paper is a preprint of a paper submitted to The 1973 National Computer Conference and Exposition

*This work was supported by the Advanced Projects Research Agency of the Department of the Defense under Contract No. DAHC15-71-C-ØØ88.

BBN Report No. 2522 March 1973

A Resource Sharing Executive For the ARPANET *

by Robert H. Thomas
Bolt, Beranek, and Newman, Inc.
Cambridge, Mass., Ø2138

* This work was supported by the Advanced Projects Research Agency of the Department of the Defense under Contract No. DAHC-71-C-ØØ88.
15

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Bolt Beranek and Newman Inc.

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP

3. REPORT TITLE
A Resource Sharing Executive for the ARPANET

4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)
Technical Report

5. AUTHOR(S) (First name, middle initial, last name)
Robert H. Thomas

6. REPORT DATE
March 1973

7a. TOTAL NO OF PAGES
39

7b. NO OF REFS
16

8a. CONTRACT OR GRANT NO
DAHC-71-C-0088
DAHC15-71-C-0088

b. PROJECT NO
ARPA ON 1967

c.

d.

9a. ORIGINATOR'S REPORT NUMBER(S)
BBN REPORT NO. 2522

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce for sale to the general public.

11. SUPPLEMENTARY NOTES
This research was sponsored by the Advanced Research Projects Agency under ARPA Order No. 1967

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

The RSEXEC (Resource Sharing Executive) is an experimental, distributed executive-like system currently operational on a number of PDP-10 (TENEX) Hosts on the ARPA computer network. The RSEXEC enlarges the range of storage and computing resources accessible to a user to include those beyond the boundaries of his local system. It includes a command language interpreter which extends the effects of user commands to include all TENEX Hosts in the ARPANET (and for certain commands some non-TENEX Hosts) and a monitor call interpreter which, in a similar manner, extends the effect of program initiated "system" calls. By acting as an intermediary between the user and non-local systems the RSEXEC removes the logical distinction between resources that are local and those that are remote. Within the RSEXEC environment neither the user nor his programs need deal directly with (or even be aware that they are dealing with) the ARPANET or the remote systems. This paper discusses considerations that led to the RSEXEC system, and describes the system in terms of the capabilities offered to the user and the approach taken to the implementation.

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
distributed computation						
computer networks						
resource sharing						
network protocols						
interprocess communication						

UNCLASSIFIED
Security Classification

TABLE OF CONTENTS

A Resource Sharing Executive for the ARPANET *

by Robert H. Thomas
Bolt, Beranek and Newman, Inc.
Cambridge, Mass. 02138

ABSTRACT

The RSEXEC (Resource Sharing Executive) is an experimental, distributed executive-like system currently operational on a number of PDP-10 (TENEX) Hosts on the ARPA computer network. The RSEXEC enlarges the range of storage and computing resources accessible to a user to include those beyond the boundaries of his local system. It includes a command language interpreter which extends the effects of user commands to include all TENEX Hosts in the ARPANET (and for certain commands some non-TENEX Hosts) and a monitor call interpreter which, in a similar manner, extends the effect of program initiated "system" calls. By acting as an intermediary between the user and non-local systems the RSEXEC removes the logical distinction between resources that are local and those that are remote. Within the RSEXEC environment neither the user nor his programs need deal directly with (or even be aware that they are dealing with) the ARPANET or the remote systems. This paper discusses considerations that led to the RSEXEC system, and describes the system in terms of the capabibities offered to the user and the approach taken to the implementation.

INTRODUCTION

The Resource Sharing Executive (RSEXEC) is a distributed, executive-like system that runs on TENEX Host computers in the ARPA computer network. The RSEXEC creates an environment which facilitates the sharing of resources among Hosts on the ARPANET. The large Hosts, by making a small amount of their resources available to small Hosts, can help the smaller Hosts provide services which would otherwise exceed their limited capacity. By sharing resources among themselves the large Hosts can provide a level of service better than any one of them could provide individually. Within the environment provided by the RSEXEC a user need not concern himself directly with network details such as communication protocols nor even be aware that he is dealing with a network.

A few facts about the ARPANET and the TENEX operating system should provide sufficient background for the remainder of this paper. Readers interested in learning more about the network or TENEX are referred to the literature; for the ARPANET references [1,2,3,4]; for TENEX, [5,6,7].

The ARPANET is a nationwide heterogeneous collection of Host computers at geographically separated locations. The Hosts differ from one another in manufacture, size, speed, word length and operating system. Communication between the

Host computers is provided by a subnetwork of small, general purpose computers called Interface Message Processors or IMPs which are interconnected by 50 kilobit common carrier lines. The IMPS are programmed to implement a store and forward communication network. As of January 1973 there were 45 Hosts on the ARPANET and 33 IMPs in the subnet.

In terms of numbers, the two most common Hosts in the ARPANET are Terminal IMPs called TIPs (12) and TENEXs (9). TIPs [8,9] are mini-Hosts designed to provide inexpensive terminal access to other network Hosts. The TIP is implemented as a hardware and software augmentation of the IMP.

TENEX is a time-shared operating system developed by BBN to run on a DEC PDP-10 processor augmented with paging hardware. In comparison to the TIPs, the TENEX Hosts are large. TENEX implements a virtual processor with a large (256K word), paged virtual memory for each user process. In addition, it provides a multi-process job structure with software program interrupt capabilities, an interactive and carefully engineered command language (implemented by the TENEX EXEC) and advanced file handling capabilities.

Development of the RSEXEC was motivated initially by the desire to pool the computing and storage resources of the individual TENEX Hosts on the ARPANET. We observed that the TENEX virtual machine was becoming a popular network resource. Further, we observed that for many users, in particular those whose access to the network is through TIPs or other non-TENEX Hosts, it shouldn't really matter which Host provides the TENEX virtual machine as long as the user is able to do his computing in the manner he has become accustomed*. A number of advantages result from such resource sharing. The user would see TENEX as a much more accessible and reliable resource. Because he would no longer be dependent upon a single Host for his computing he would be able to access a TENEX virtual machine even when one or more of the TENEX Hosts were down. Of course, for him to be able to do so in a useful way, the TENEX file system would have to span across Host boundaries. The individual TENEX Hosts would see advantages also. At present, due to local storage limitations, some sites do not provide all of the TENEX subsystems to their users. For example, one site doesn't support FORTRAN for this reason. Because the subsystems available would, in effect, be the "union" of the subsystems available on all TENEX Hosts, such Hosts would be able to provide access to all TENEX subsystems.

---

* This, of course, ignores the problem of differences in the accounting and billing practices of the various TENEX Hosts. Because all of the TENEX Hosts (with the exception of the two at BBN) belong to ARPA we felt that the

administrative problems could be overcome if the technical problems preventing resource sharing were solved.

The RSEXEC was conceived of as an experiment to investigate the feasibility of the multi-Host TENEX concept. Our experimentation with an initial version of the RSEXEC was encouraging and, as a result, we planned to develop and maintain the RSEXEC as a TENEX subsystem. The RSEXEC is, by design, an evolutionary system; we planned first to implement a system with limited capabilities and then to let it evolve, expanding its capabilities, as we gained experience and came to understand the problems involved.

During the early design and implementation stages it became clear that certain of the capabilities planned for the RSEXEC would be useful to all network users, as well as users of a multi-Host TENEX. The ability of a user to inquire where in the network another user is and then to "link" his own terminal to that of the other user in order to engage in an on-line dialogue is an example of such a capability.

A large class of users with a particular need for such capabilities are those whose access to the network is through mini-Hosts such as the TIP. At present TIP users account for a significant amount of network traffic, approximately 35% on an average day [10]. A frequent source of complaints by TIP users is the absence of a sophisticated command language interpreter for TIPs and, as a result, their inability to obtain information about network status, the status of various Hosts, the wherabouts of other users, etc., without first logging into some Host. Furthermore, even after they log into a Host, the information readily available is generally limited to the Host they log into. A command language interpreter of the type desired would require more (core memory) resources than are available in a TIP alone. We felt that with a little help from one or more of the larger Hosts it would be feasible to provide TIP users with a good command language interpreter. (The TIPs were already using the storage resources of one TENEX Host to provide their users with a network news service [10,11].) Further, since a subset of the features already planned for the RSEXEC matched the needs of the TIP users, it was clear that with little additional effort the RSEXEC system could provide TIP users with the command language interpreter they needed. The service TIP users can obtain through the RSEXEC by the use of a small portion of the resources of several network Hosts is superior to that they could obtain either from the TIP itself or from any single Host.

An initial release of the RSEXEC as a TENEX subsystem has been distributed to the ARPANET TENEX Hosts. In addition, the RSEXEC is available to TIP users (as well as other network users) for use as a network command language interpreter, preparatory to logging into a particular Host (of course, if the user chooses to log into TENEX he may continue using the RSEXEC after login). Several non-TENEX Hosts have expressed interest in the RSEXEC system, particularly in the capabilities it supports for inter-Host user-user interaction, and these Hosts are now participating in the RSEXEC experiment.

The current interest in computer networks and their potential for resource sharing suggests that other systems similar to the RSEXEC will be developed. At present there is relatively little in the literature describing such distributing computing systems. This paper is presented to record our experience with one such system; we hope it will be useful to others considering the implementation of such systems.

The remainder of this paper describes the RSEXEC system in more detail: first, in terms of what the RSEXEC user sees, and then, in terms of the implementation.

THE USER'S VIEW OF THE RSEXEC

The RSEXEC enlarges the range of storage and computing resources accessible to a user to include those beyond the boundaries of his local system. It does that by making resources, local and remote, available as part of a single, uniformly accessible pool. The RSEXEC system includes a command language interpreter which extends the effect of user commands to include all TENEX Hosts in the ARPANET (and for certain commands some non-TENEX Hosts), and a monitor call interpreter which, in a similar way, extends the effect of program initiated "system" calls.

To a large degree the RSEXEC relieves the user and his programs of the need to deal directly with (or even be aware that they are dealing with) the ARPANET or remote Hosts. By acting as an intermediary between its user and non-local Hosts the RSEXEC removes the logical distinction between resources that are local and those that are remote. In many contexts references to files and devices* may be made in a site independent manner. For example, although his files may be distributed among several Hosts in the network, a user need not specify where a particular file is stored in order to delete it; rather, he need only supply the file's name to the delete command.

* Within TENEX, peripheral devices are accessible to users

via the file system; the terms "file" and "device" are frequently used interchangeably in the following.

To a first approximation, the user interacts with the RSEXEC in much the same way as he would normally interact with the standard (single Host) TENEX executive program. The RSEXEC command language is syntactically similar to that of the EXEC. The significant difference, of course, is a semantic one; the effect of commands are no longer limited to just a single Host.

Some RSEXEC commands make direct reference to the multi-Host environment. The facilities for inter-Host user-user interaction are representative of these commands. For example, the WHERE and LINK commands can be used to initiate an on-line dialogue with another user:

```
←WHERE (IS USER) JONES *
   JOB 17 TTY6 USC
   JOB 5 TTY14 CASE
←LINK (TO TTY) 14 (AT SITE) CASE
```

* "←" is the RSEXEC "ready" character. The words enclosed in parentheses are "noise" words which serve to make the commands more understandable to the user and may be omitted. A novice user can use the character ESC to cause the RSEXEC to prompt him by printing the noise words.

Facilities such as these play an important role in removing the distinction between "local" and "remote" by allowing users of geographically separated Hosts to interact with one another as if they were members of a single user community. The RSEXEC commands directly available to TIP users in a "pre-login state" include those for inter-Host user-user interaction together with ones that provide Host and network status information and network news.

Certain RSEXEC commands are used to define the "configuration" of the multi-Host environment seen by the user. These "meta" commands enable the user to specify the "scope" of his subsequent commands. For example, one such command (described in more detail below) allows him to enlarge or reduce the range of Hosts encompassed by file system commands that follow. Another "meta" command enables him to specify a set of peripheral devices which he may reference in a site independent manner in subsequent commands.

The usefulness of multi-Host systems such as the RSEXEC is, to a large extent, determined by the ease with which a user can manipulate his files. Because the Host used one day may be different from the one used the next, it is <u>necessary</u> that a user be able to reference any given file from all Hosts. Furthermore, it is desirable that he be able to reference the file in the same manner from all Hosts.

The file handling facilities of the RSEXEC were designed to:

1. make it possible to reference any file on any Host by implementing a file name space which spans across Host boundaries.
2. make it convenient to reference frequently used files by supporting "short hand" file naming conventions, such as the ability to specify certain files without site qualification.

The file system capabilities of the RSEXEC are designed to be available to the user at the command language level and to his programs at the monitor call level. An important design criterion was that existing programs be able to run under the RSEXEC without reprogramming.

File access within the RSEXEC system can be best described in terms of the commonly used model which views the files accessible from within a Host as being located at terminal nodes of a tree. Any file can be specified by a pathname which describes a path through the tree to the file. The complete pathname for a file includes every branch on the path leading from the root node to the file. While, in general, it is necessary to specify a complete pathname to uniquely identify a file, in many situations it is possible to establish contexts within which a partial pathname is sufficient to uniquely identify a file. Most operating systems provide such contexts, designed to allow use of partial pathnames for frequently referenced file, for their users*.

It is straightforward to extend the tree structured model for file access within a single Host to file access within the entire network. A new root node is created with branches to each of the root nodes of the access trees for the individual Hosts, and the complete pathname is enlarged to include the Host name. A file access tree for a single Host is shown in Figure 1; Figure 2 shows the file access

* For example, TENEX does it by:

1. Assuming default values for certain components left unspecified in partial pathnames;
2. Providing a reference point for the user within the tree (working directory) and thereafter interpreting partial pathnames as being relative to that point. TENEX sets the reference point for each user at login time and, subject to access control restrictions, allows the user to change it (by "connecting" to another directory).

tree for the network as a collection of single Host trees.

The RSEXEC supports use of complete pathnames that include a Host component thereby making it possible (albeit somewhat tedious) for users to reference a file on any Host. For example, the effect of the command

```
←APPEND (FILE) [CASE]DSK:<THOMAS>DATA.NEW
        (TO FILE) [BBN]DSK:<BOBT>DATA.OLD  *
```

is to modify the file designated ① in Figure 2 by appending to it the file designated ②.

To make it convenient to reference files, the RSEXEC allows a user to establish contexts for partial pathname interpretation. Since these contexts may span across several Hosts, the user has the ability to configure his own "virtual" TENEX which may in reality be realized by the resources of several TENEXs. Two mechanisms are available to do this.

* The syntax for (single host) TENEX pathnames includes device, directory, name and extension components. The RSEXEC extends that syntax to include a Host component. The pathname for ② specifies: the CASE Host; the disk ("DSK") device; the directory THOMAS; the name DATA; and the extension NEW.

The first of these mechanisms is the user profile which is a collection of user specific information and parameters maintained by the RSEXEC for each user. Among other things, a user's profile specifies a group of file directories which taken together define a composite directory for the user. The "contents" of the composite directory are the union of the "contents" of the file directories specified in the profile. When a pathname without site and directory qualification is used, it is interpreted relative to the user's composite directory. The composite directory serves to define a reference point within the file access tree that is used by the RSEXEC to interpret partial pathnames. That reference point is somewhat unusual in that it spans several Hosts.

One of the ways a user can reconfigure his "virtual" TENEX is by editing his profile. With one of the "meta" commands noted earlier he can can add or remove components of his composite directory to control how partial pathnames are interpreted.

An example may help clarify the role of the user profile, the composite directory and profile editing. Assumn that the profile for user Thomas contains directories BOBT at BBN, THOMAS at CASE and BTHOMAS at USC (see Figure 2). His composite directory, the reference point for pathname interpretation, spans three Hosts. The command

```
←APPEND (FILE) DATA.NEW (TO FILE) DATA.OLD
```

achieves the same effect as the APPEND command in a previous example. To respond the RSEXEC first consults the composite directory to discover the locations of the files, and then acts to append the first file to the second; how it does so is discussed in the next section. If he wanted to change the scope of partial pathnames he uses, user Thomas could delete directory BOBT at BBN from his profile and add directory RHT at AMES to it.

The other mechanism for controlling the interpretation of partial pathnames is device binding. A user can instruct the RSEXEC to interpret subsequent use of a particular device name as referring to a device at the Host he specifies. After a device name has been bound to a Host in this manner, a partial pathname without site qualification that includes it is interpreted as meaning the named device at the specified Host. Information in the user profile specifies a set of default device bindings for the user. The binding of devices can be changed dynamically during an RSEXEC session. In the context of the previous example the sequence of commands:

```
←BIND (DEVICE) LPT (TO SITE) BBN
←LIST DATA.NEW
←BIND (DEVICE) LPT (TO SITE) USC
←LIST DATA.NEW
```

produces two listings of the file DATA.NEW: one on the line printer (device "LPT") at BBN, the other on the printer at

USC. As with other RSEXEC features, device binding is available at the program level. For example, a program that reads from magnetic tape will function properly under the RSEXEC when it runs on a Host without a local mag-tape unit, provided the mag-tape device has been bound properly.

The user can take advantage of the distributed nature of the file system to increase the "accessibility" of certain files he considers important by instructing the RSEXEC to maintain images of them at several diferent Hosts. With the exception of certain special purpose files (e.g., the user's "message" file), the RSEXEC treats files with the same pathname relative to a user's composite directory as images of the same multi-image file. The user profile is implemented as a multi-image file with an image maintained at every component directory of the composite directory*.

* The profile is somewhat special in that it is accessible to the user only through the profile editing commands, and is otherwise transparent.

IMPLEMENTATION OF THE RSEXEC

The RSEXEC implementation is discussed in this section with the focus on approach rather than detail. The result is a simplified but nonetheless accurate sketch of the implementation.

The RSEXEC system is implemented by a collection of programs which run with no special privileges on TENEX Hosts. The advantage of a "user-code" (rather than "monitor-code") implementation is that ordinary user access is all that is required at the various Hosts to develop, debug and use the system. Thus experimentation with the RSEXEC can be conducted with minimal disruption to the TENEX Hosts.

The ability of the RSEXEC to respond properly to users' requests often requires cooperation from one or more remote Hosts. When such cooperation is necessary, the RSEXEC program interacts with RSEXEC "service" programs at the remote Hosts according to a pre-agreed upon set of conventions or protocol. Observing the protocol, the RSEXEC can instruct a service program to perform actions on its behalf to satisfy its user's requests.

Each Host in the RSEXEC system runs the service program as a "demon" process which is prepared to provide service to any remote process that observes protocol. The relation between RSEXEC programs and these demons is shown schematically in Figure 3.

The RSEXEC Protocol

The RSEXEC protocol is a set of conventions designed to support the interprocess communicaton requirements of the RSEXEC system. The needs of the system required that the protocol:

1. be extensible:

   As noted earlier, the RSEXEC is, by design, an evolutionary system.

2. support many-party as well as two-party interactions:

   Some situations are better handled by single multi-party interactions than by several two-party interactions. Response to an APPEND command when the files and the RSEXEC are all at different Hosts is an example (see below).

3. be convenient for interaction between processes running on dissimilar Hosts while supporting efficient interaction between processes on similar Hosts:

   Many capabilities of the RSEXEC are useful to users of non-TENEX as well as TENEX Hosts. It is important that the protocol not favor TENEX at the expense of other Hosts.

The RSEXEC protocol has two parts:

1. a protocol for initial connection specifies how programs desiring service (users) can connect to programs providing service (servers);
2. a command protocol specifies how the user program talks to the server program to get service after it is connected.

The protocol used for initial connection is the standard ARPANET initial connection protocol (ICP) [12]. The communication paths that result from the ICP exchange are used to carry commands and responses between user and server. The protocol supports many-party interaction by providing for the use of auxillary communication paths, in addition to the command paths. Auxillary paths can be established at the user's request between server and user or between server and a third party. Communication between processes on dissimilar Hosts usually requires varying degrees of attention to message rormatting, code conversion, byte manipulation, etc. The protocol addresses the issue of convenience in the way other standard ARPANET protocols have [13,14,15]. It specifies a default message format designed to be "fair" in the sense that it doesn't favor one type of Host over another by requiring all reformatting be done by one type of Host. It addresses the issue of efficiency by providing a mechanism with which processes on similar Hosts can negotiate a change in format from the default to one better suited for efficient use by their Hosts.

The protocol can perhaps best be explained further by examples that illustrate how the RSEXEC uses it. The following discusses its use in the WHERE, APPEND and LINK commands:

←WHERE (IS USER) JONES

The RSEXEC queries each non-local server program about user Jones. To query a server, it establishs connections with the server; transmits a "request for information about Jones" as specified by the protocol; and reads the response which indicates whether or not Jones is a known user, and if he is, the status of his active jobs (if any).

←APPEND (FILE) DATA.NEW (TO FILE) DATA.OLD

Recall that the files DATA.NEW and DATA.OLD are at CASE and BBN, respectively; assume that the APPEND request is made to an RSEXEC running at USC. The RSEXEC connects to the servers at CASE and BBN. Next, using the appropriate protocol commands, it instructs each to establish an auxilary path to the other (see Figure 4). Finally, it instructs the server at CASE to transmit the file DATA.NEW over the auxillary connection and the server at BBN to append the data it reads from the auxiliary connection to the file DATA.OLD.

<LINK (TO TTY) 14 (AT SITE) CASE

Assume that the user making the request is at USC. After connecting to the CASE server, the RSEXEC uses appropriate protocol commands to establish two auxillary connections (one "send" and one "receive") with the server. It next instructs the server to "link" its (the server's) end of the auxillary connections to Terminal 14 at its (the server's) site. Finally, to complete the LINK command the RSEXEC "links" its end of the auxillary connections to its user's terminal.

The RSEXEC Program

A large part of what the RSEXEC program does is to locate the resources necessary to satisfy user requests. It can satisfy some requests directly whereas others may require interaction with one or more remote server programs. For example, an APPEND command may involve interaction with none, one or two server programs depending upon where the two files are stored.

A issue basic to the RSEXEC implementation concerns handling information necessary to access files: in particular, how much information about non-local files should be maintained locally by the RSEXEC? The advantage of maintaining the information locally is that requests requiring it can be satisfied without incurring the overhead involved in first locating the information and then accessing it through the network. Certain highly interactive activity would be precluded if it required significant interaction with remote server programs. For example, recognition and completion of file names* would be unusable if it required direct interaction with several remote server programs. Of course, it would be impractical to maintain information locally about all files at all TENEX Hosts.

The approach taken by the RSEXEC is to maintain information about the non-local files a user is most likely to reference and to acquire information about others from remote server programs as necessary. It implements this

* File name recognition and completion is a TENEX feature which allows a user to abbreviate fields of a file pathname. Appearance of ESC in the name causes the portion of the field before the ESC to be looked up, and, if the portion is unambiguous, the system will recognize it and supply the omitted characters and/or fields to complete the file name. If the portion is ambiguous, the system will prompt the user for more characters by ringing the terminal bell. Because of its popularity we felt it important that the RSEXEC support this feature.

strategy by distinguishing internally four file types:

1. files in the Composite Directory;
2. files resident at the local Host which are not in the Composite Directory;
3. files accessible via a bound device, and;
4. all other files.

Information about files of type 1 and 3 is maintained locally by the RSEXEC. It can acquire information about type 2 files directly from the local TENEX monitor, as necessary. No information about type 4 files is maintained locally; whenever such information is needed it is acquired from the appropriate remote server. File name recognition and completion and the use of partial pathnames is restricted to file types 1, 2 and 3.

The composite directory contains an entry for each file in each of the component directories specified in the user's profile. At the start of each session the RSEXEC constructs the user's composite directory by gathering information from the server programs at the Hosts specified in the user profile. Throughout the session the RSEXEC modifies the composite directory, adding and deleting entries, as necessary. The composite directory contains frequently accessed information (e.g., Host location, size, date of last access, etc.) about the user's files. It represents a source of information that can be accessed without incurring the overhead of going to the remote Host each time it is needed.

THE RSEXEC regards the composite directory as an approximation (which is usually accurate) to the state of the user's files. The state of a given file is understood to be maintained by the TENEX monitor at the site where the file resides. The RSEXEC is aware that the outcome of any action it initiates involving a remote file depends upon the file's state as determined by the appropriate remote TENEX monitor, and that the state information in the composite directory may be "out of phase" with the actual state. It is prepared to handle the occassional failure of actions it initiates based on inaccurate information in the composite directory by giving the user an appropriate error message and updating the composite directory. Depending upon the severity of the situation it may choose to change a single entry in the composite directory, reacquire all the information for a component directory, or rebuild the entire composite directory.

The Service Program for the RSEXEC

Each RSEXEC service program has two primary responsibilities:

1. to act on behalf of non-local users (typically RSEXEC programs), and;
2. to maintain information on the status of the other server programs.

The status information it maintains has an entry for each Host indicating whether the server program at the Host is up and running, the current system load at the Host, etc. Whenever an RSEXEC program needs service from some remote server program it checks the status information maintained by the local server. If the remote server is indicated as up it goes ahead and requests the service; otherwise it does not bother.

A major requirement of the server program implementation is that it be resilient to failure. The server should be able to recover gracefully from common error situations and, more important, it should be able to "localize" the effects of those from which it can't. At any given time, the server may simultaneously be acting on behalf of a number of user programs at different Hosts. A malfunctioning or malicious user program should not be able to force termination of the entire service program. Further, it should not be able to adversely effect the quality of service received by the other users.

To achieve such resiliency the RSEXEC server program is implemented as a hierarchy of loosely connected, cooperating processes (see Figure 5):

1. The RSSER process is at the root of the hierarchy. Its primary duty is to create and maintain the other processes;

2. REQSER processes are created in response to requests for service. There is one for each non-local user being served.
3. A STASER process maintains status information about the server programs at other sites.

Partitioning the server in this way makes it easy to localize the effect of error situations. For example, occurrence of an unrecoverable error in a REQSER process results in service interruption only to the user being serviced by that process: all other REQSER processes can continue to provide service uninterrupted.

When service is requested by a non-local program, the RSSER process creates a REQSER process to provide it. The REQSER process responds to requests by the non-local program as governed by the protocol. When the non-local program signals that it needs no further service, the REQSER process halts and is terminated by RSSER.

The STASER process maintains an up-to-date record of the status of the server programs at other Hosts by exchanging status information with the STASER processes at the other Hosts. The most straightforward way to keep up-to-date information would be to have each STASER process periodically "broadcast" its own status to the others. Unfortunately, the current, connection-based Host-Host protocol of the ARPANET [16] forces use of a less elegant mechanism. Each STASER process performs its task by:

1. periodically requesting a status report from each of the other processes, and;
2. sending status information to the other processes as requested.

To request a status report from another STASER process, STASER attempts to establish a connection to a "well known" port maintained in a "listening" state by the other process. If the other process is up and running, the connection attempt succeeds and status information is sent to the requesting process. The reporting process then returns the well-known port to the listening state so that it can respond to requests from other processs. The requesting process uses the status report to update an appropriate status table entry. If the connection attempt does not succeed within a specified time period, the requesting process records the event as a missed report in an appropriate status table entry.

When the server program at a Host first comes up, the status table is initialized by marking the server programs at the other Hosts as down. After a particular server is marked as down, STASER must collect a number of status reports from it before it can mark the program as up and useful. If, on its way up, the program misses several consecutive reports, its "report count" is zeroed. By requiring a number of status reports from a remote server before marking it as up, STASER is requiring that the remote program has functioned "properly" for a while. As a result, the likelihood that it is in a stable state capable of servicing local RSEXEC programs is increased. STASER is willing to attribute occassional missed reports as being due to "random" fluctuations in network or Host responses. However, consistent failure of a remote server to report is taken to mean that the program is unusable and results in it being marked as down.

Because up-to-date status information is crucial to the operation of the RSEXEC system it is important that failure of the STASER process be infrequent, and that when a failure does occur it is detected and corrected quickly. STASER itself is programmed to cope with common errors. However error situations can arise from which STASER is incapable of recovering. These situations are usually the result of infrequent and unexpected "network" events such as Host-Host protocol violations and lost or garbled messages. (Error detection and control is performed on messages passed between IMPS to insure that messages are not lost or garbled within the IMP subnet; however, there is currently no error control for messages passing over the Host to IMP interface.) For all practical purposes such situations are irreproducible, making their patholcgy difficult to understand let alone program for. The approach we have taken is to acknowledge that we don't know how to prevent such situations and to try to minimize their effect. When functioning properly the STASER process "reports in" periodically. If it fails to report as expected, RSSER assumes that it has malfunctioned and restarts it.

Providing the RSEXEC to TIP Users

The RSEXEC is available as a network executive program to users whose access to the network is by way of a TIP (or other non-TENEX Host) through a standard service program (TIPSER) that runs on TENEX Hosts*. To use the RSEXEC from a TIP a user instructs the TIP to initiate an initial connection protocol exchange with one of the TIPSER programs. TIPSER responds to the ICP by creating a new process which runs the RSEXEC for the TIP user.

---

* At present TIPSER is run on a regular basis at only one of the TENEX Hosts; we expect several other Hosts will start running it on a regular basis shortly.

CONCLUDING REMARKS

Experience with the RSEXEC has shown that it is capable of supporting significant resource sharing among the TENEX Hosts in the ARPANET. It does so in a way that provides users access to resources beyond the boundaries of their local system with a convenience not previously experienced within the ARPANET. As the RSEXEC system evolves, the TENEX Hosts will become more tightly coupled and will approach the goal of a multi-Host TENEX. Part of the process of evolution will be to provide direct support for many RSEXEC features at the level of the TENEX monitor.

At present the RSEXEC system is markedly deficient in supporting significant resource sharing among dissimilar Hosts. True, it provides mini-Hosts, such as TIPs, with a mechanism for accessing a small portion of the resources of the TENEX (and some non-TENEX) Hosts, enabling them to provide their users with an executive program that is well beyond their own limited capacity. Beyond that, however, the system does little more than to support inter-Host user-user interaction between Hosts that choose to implement the appropriate subset of the RSEXEC protocol. There are, of course, limitations to how tightly Hosts with fundamentally different operating systems can be coupled. However, it is clear that the RSEXEC has not yet approached those limitations and that there is room for improvement in this area.

The RSEXEC is designed to provide access to the resources within a computer network in a manner that makes the network itself transparent by removing the logical distinction between local and remote. As a result, the user can deal with the network as a single entity rather than a collection of autonomous Hosts. We feel that it will be through systems such as the RSEXEC that users will be able to most effectively exploit the resources of computer networks.

ACKNOWLEDGEMENTS

Apreciation is due to W.R. Sutherland whose leadership and enthusiasm made the RSEXEC project possible. P.R. Johnson actively contributed in the implementaion of the RSEXEC. The TENEX Group at BBN deserves recognition for constructing an operating system that made the task of implementing the RSEXEC a pleasant one.

REFERENCES

1 L G Roberts, B D Wessler

Computer network development to achieve resource sharing

Proceedings of AFIPS SJCC 1970, Vol 36, p543-549.

2 F E Heart, R E Kahn, S M Ornstein, W R Crowther, D C Walden

The interface message processor for the ARPA computer network

Proceedings of AFIPS SJCC 1970, Vol 36.

3 J M McQuillan, W R Crowther, B P Cosell, D C Walden, F E Heart

Improvements in the design and performance of the ARPA network

Proceedings of AFIPS FJCC 1972, Vol 41, p741-754.

4 L G Roberts

A forward look

SIGNAL Vol XXV, No. 12, p77-81, Aug 1971.

5 D G Bobrow, J D Burchfiel, D L Murphy, R S Tomlinson

TENEX, a paged time sharing system for the PDP-10

Communications of the ACM, Vol 15, No 3, p135-143, March 1972.

6 BBN Computer Science Division

TENEX JSYS manual - a manual of TENEX monitor calls

BBN Cambridge Mass., Nov 1971.

7 D L Murphy
Storage organization and management in TENEX
Proceedings of AFIPS FJCC 1972, Vol 41, p23-32.

8 S M Ornstein, F E Heart, W R Crowther, H K Rising, S B Russell, A Michel
The terminal IMP for the ARPA computer network
Proceedings of AFIPS 1972 SJCC, Vol 40, p243-254.

9 R E Kahn
Terminal access to the ARPA computer network
Courant Computer Symposium 3 - Computer Networks
Courant Institue, New York, Nov 1970.

10 N W Mimno, B P Cosell, D C Walden, S C Butterfield, J B Levin
Terminal access to the ARPA network: experience and improvement
Proceedings COMPCON '73, Seventh Annual IEEE Computer Society International Conference.

11 D C Walden
TIP user's guide
BBN Report No 2183, Sept 1972; also available from the Network Information Center at Stanford Research Institute, Menlo Park California, as Document NIC# 10916.

12 J B Postel

Official initial connection protocol

Available from Network Information Center as Document NIC# 71Ø1.

13 J B Postel

TELNET protocol

ARPA Network Working Group Request For Comments #318, Available from Network Information Center as Document NIC# 9348.

14 A K Bhushan

File transfer protocol

ARPA Network Working Group Request For Comments #358, Available from Network Information Center as Document NIC# 1Ø596.

15 S D Crocker, J F Heafner, R M Metcalfe, J B Postel

Function oriented protocols for the ARPA computer network

Proceedings of AFIPS 1972 SJCC, Vol 4Ø, p271-279.

16 A McKenzie

Host/host protocol for the ARPA network

Available from the Network Information Center as Document NIC# 8246.

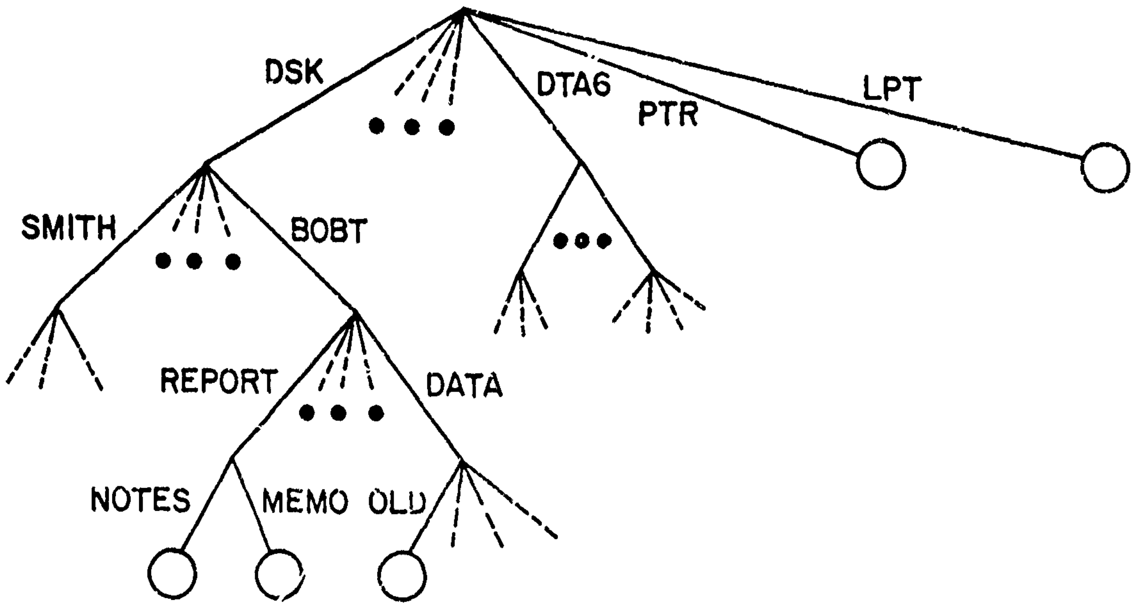

Figure 1 File access tree for a single Host. The circles at the terminal nodes of the tree represent files.

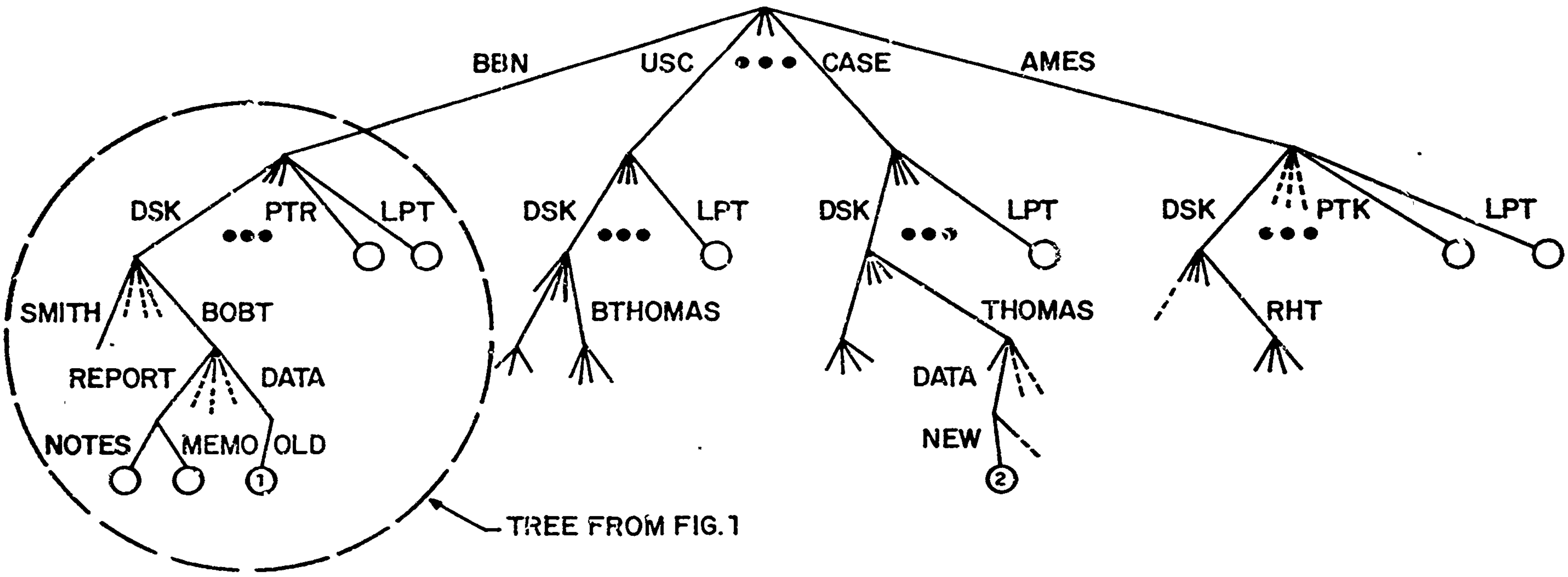

Figure 2 File access tree for a network. The single Host access tree from Figure 1 is part of this tree.

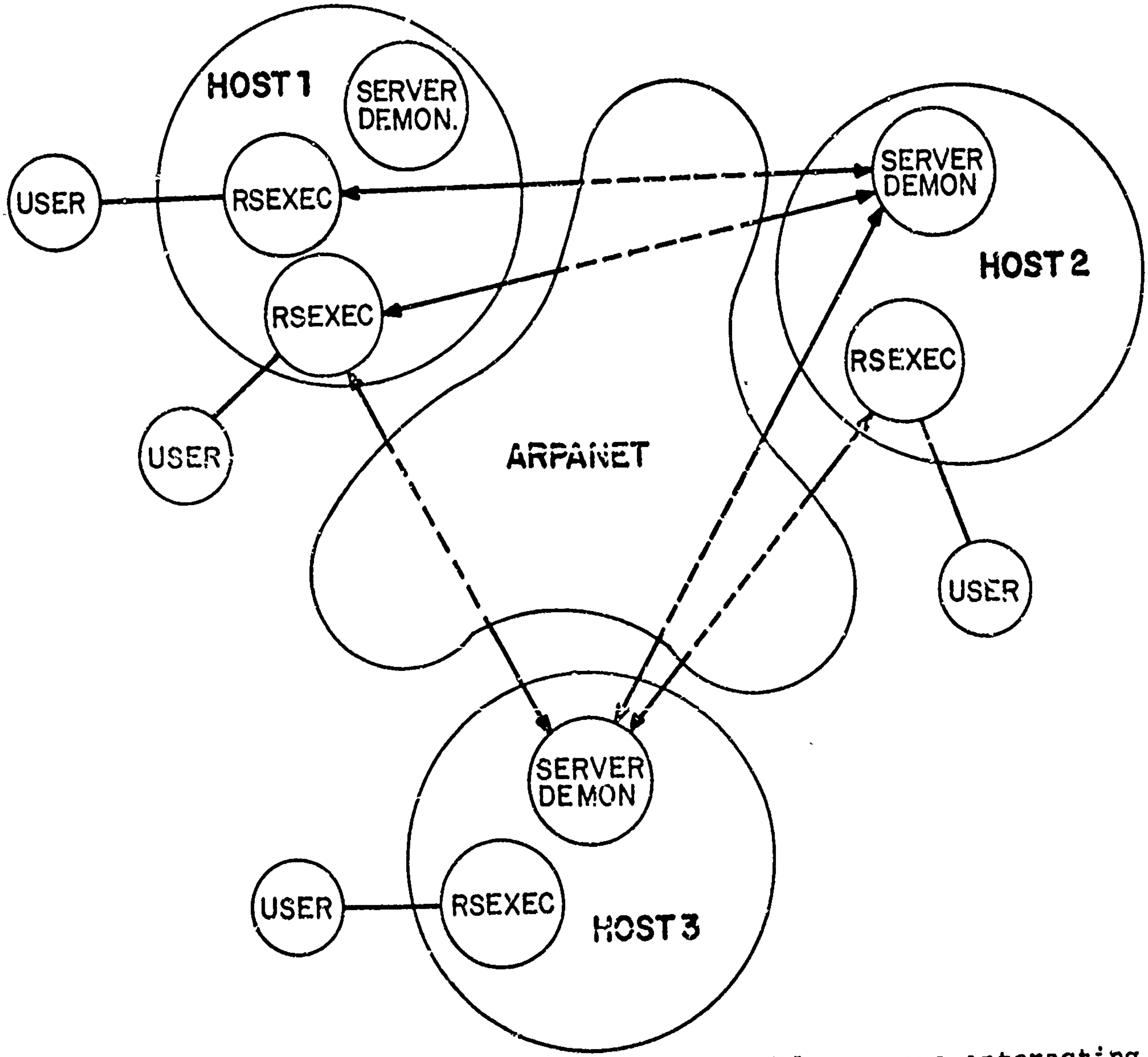

Figure 3 Schematic showing several RSEXEC programs interacting, on behalf of their users, with remote server programs.

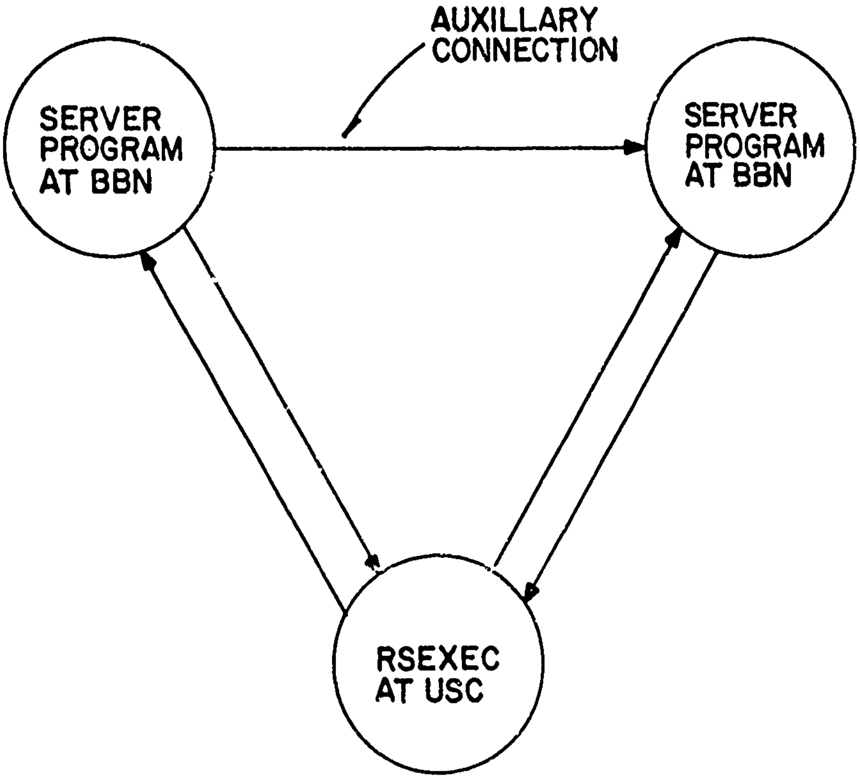

Figure 4 Configuration of RSEXEC and two server programs required to satisfy an APPEND command when the two files and the RSEXEC are all on different Hosts. The auxiliary connection is used to transmit the file to be appended from one server to the other.

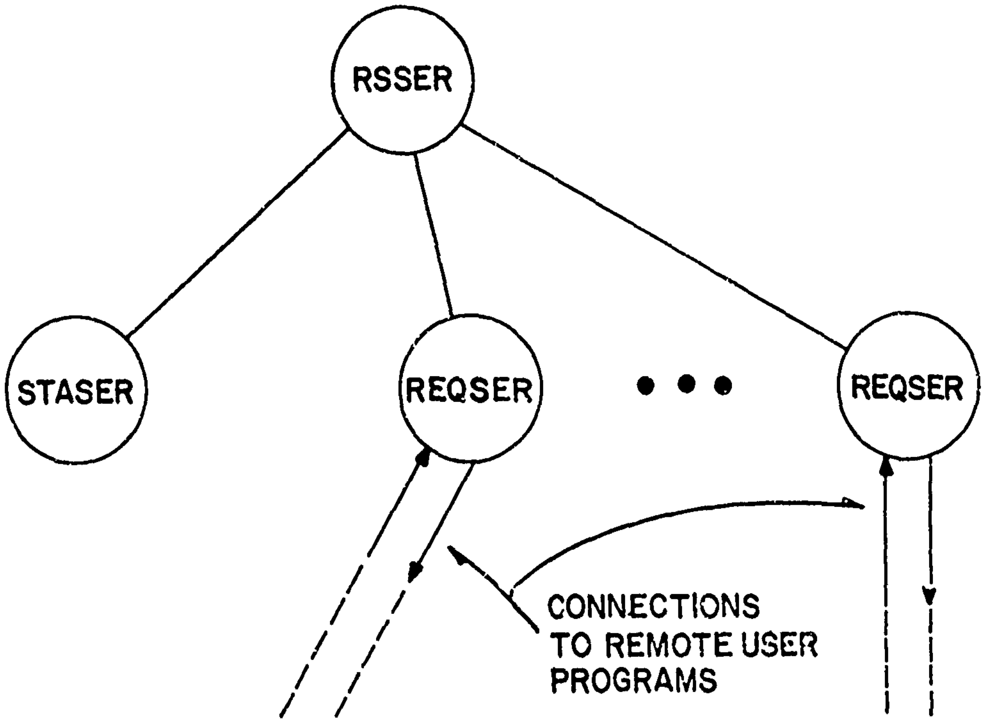

Figure 5 Hierarchical structure of the RSEXEC service program.